

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH

---∞📖∞---



**BÁO CÁO ĐỀ TÀI MÔN PHÁT TRIỂN ỨNG DỤNG
CHO CÁC THIẾT BỊ DI ĐỘNG**

ĐỀ TÀI: ỨNG DỤNG BÁO THỨC

Giáo viên hướng dẫn:	GV. Trương Bá Thái
Sinh viên thực hiện :	Đào Phi Lạc (N15DCCN005) Nguyễn Anh Nhật (N15DCCN37) Mạc Đỗ Xuân Tính (N15DCCN039) Trần Thị Thanh (N15DCCN046) Nguyễn Đào Hồng Nhung (N15DCCN065)
Lớp :	D15CQCP01-N
Nhóm :	14
Đề tài :	Ứng dụng Báo Thức

Thành phố Hồ Chí Minh, 2019

Chương 1: Giới thiệu	3
I. Đề tài:	3
1. Tên đề tài:	3
2. Mô tả:	3
II. Cơ sở dữ liệu:	3
Chương 2: Phân tích và thiết kế hệ thống	3
I. SRS (Software Requirements Specification)	3
1. Màn hình chính:	3
2. Màn hình cài đặt của app:	4
3. Màn hình cài đặt của từng báo thức:	4
4. Màn hình chọn nhạc chuông:	6
5. Màn hình chọn cách tắt báo thức:	7
6. Màn hình cài đặt chế độ lắc:	8
7. Màn hình thử thách lắc:	8
8. Màn hình cài đặt chế độ làm toán:	9
9. Màn hình thử thách làm toán:	10
10. Màn hình cài đặt thử thách di chuyển	10
11. Màn hình thử thách di chuyển	11
II. SDS (Software Design Specification)	12
1. Màn hình chính:	12
2. Màn hình cài đặt của app:	12
3. Màn hình cài đặt của từng báo thức:	12
4. Màn hình chọn nhạc chuông:	13
5. Màn hình chọn cách tắt báo thức:	13
6. Màn hình cài đặt chế độ lắc:	13
7. Màn hình thử thách lắc:	14
8. Màn hình cài đặt chế độ làm toán:	14
9. Màn hình thử thách làm toán:	14
10. Màn hình cài đặt thử thách di chuyển	14
11. Màn hình thử thách di chuyển:	15
Chương 3: Cài đặt và kiểm thử	15
1. Màn hình chính:	15
a. Màn hình chính: MainActivity	15
b. Màn hình các báo thức: UpcomingAlarmFragment.	16
c. AlarmAdapter:	18

2.	Màn hình cài đặt của app:	20
a.	Thời gian im lặng của báo thức	20
b.	Số lần im lặng của báo thức	20
c.	Tự động tắt báo thức	20
d.	Nút Back: quay lại màn hình chính	21
e.	Check box Tăng dần âm lượng báo thức	21
f.	Check box Ngăn tắt nguồn điện thoại khi báo thức reo	21
g.	Radio button chọn cách hiển thị giờ (24 or 12)	21
h.	Hiệu ứng animation: fade in và blink	21
3.	Màn hình cài đặt của từng báo thức:	21
4.	Màn hình chọn nhạc chuông:	22
5.	Màn hình chọn cách tắt báo thức:	23
6.	Màn hình cài đặt chế độ lắc:	26
a.	Lựa chọn thêm hoặc giảm số lần lắc	26
b.	Chọn mức độ lắc: Dễ, vừa, khó	27
c.	Hiển thị số lần lắc điện thoại tại text view Shake for...times	27
7.	Màn hình thử thách lắc:	27
8.	Màn hình cài đặt chế độ làm toán:	28
a.	Set các giá trị cho TextView số phép toán và ví dụ mẫu cho mức độ khó đã chọn qua radio button.	28
b.	Xác nhận, Hủy bỏ tùy chọn:	29
c.	Cài đặt mặc định cho báo thức:	29
9.	Màn hình thử thách làm toán:	30
a.	Tạo phép tính mới:	30
b.	Xác nhận, Xóa kết quả vừa nhập:	31
c.	Animation:	31
10.	Màn hình cài đặt thử thách di chuyển	32
11.	Màn hình thử thách di chuyển	32
Chương 4: Kết luận		33
1.	Kết quả:	33
2.	Hạn chế:	33

Chương 1: Giới thiệu

I. Đề tài:

1. Tên đề tài:

- Ứng dụng báo thức dành cho thiết bị android (từ **Android 5.1 API 22** trở lên)

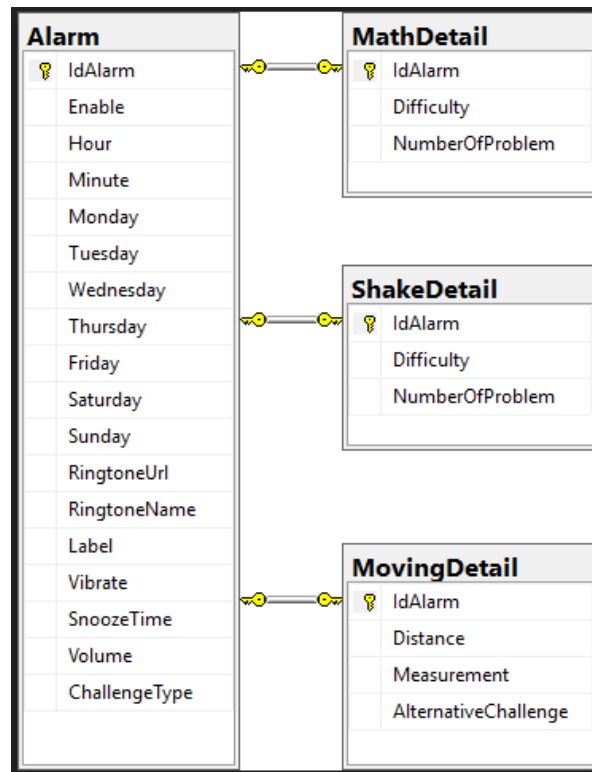
2. Mô tả:

- Ứng dụng báo thức đặc biệt với hai chế độ báo thức:

+ Báo thức mặc định: Khi báo thức reo, người dùng bấm tắt thì báo thức sẽ ngừng.

+ Báo thức đặc biệt: Khi báo thức reo, người dùng phải hoàn thành một thử thách (di chuyển, lắc điện thoại, trả lời các phép toán) khi đó báo thức mới ngừng reo.

II. Cơ sở dữ liệu:



Chương 2: Phân tích và thiết kế hệ thống

I. SRS (Software Requirements Specification)

1. Màn hình chính:







- Tên người làm: Mạc Đỗ Xuân Tính

- Hình:

- Chức năng của màn hình này:

+ Hiển thị thời gian còn lại từ hiện tại cho đến báo thức gần nhất.

+ Hiển thị danh sách các báo thức được đặt, thông tin từng báo thức.

- + Hiện thị nút Thêm báo thức, nút chuyển Màn hình báo thức, nút chuyển Màn hình cài đặt.
- Các Control+View:
 - +  TextView hiển thị thời gian còn lại từ hiện tại đến báo thức gần nhất.
 - +  Item báo thức hiển thị các thông tin thiết lập của báo thức đó.
 - +  Switch bật/tắt báo thức.
 - + RecyclerView hiển thị danh sách các báo thức đã được đặt.
 - +  Nút Thêm báo thức mới: Hiển thị màn hình thêm báo thức mới.
 - +  Nút Báo thức: Hiển thị trang màn hình các báo thức.
 - +  Nút Cài đặt: Hiển thị màn hình cài đặt của ứng dụng.

2. Màn hình cài đặt của app:

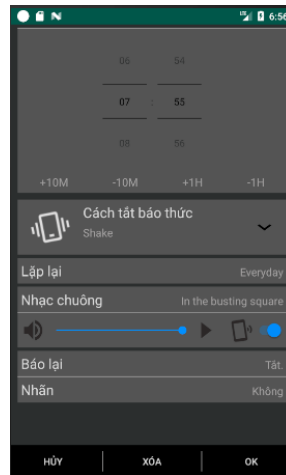
- Tên người làm: Nguyễn Đào Hồng Nhung
- Hình:



- Chức năng của màn hình này:
- Các button:
 - + Thời gian im lặng của báo thức: để im lặng khi người dùng bấm vào khi báo thức reo.
 - + Số lần im lặng: Chọn số lần cho phép im lặng của báo thức
 - + Tự động tắt báo thức: Tắt báo thức nếu reo quá thời gian.
 - + Back: quay lại màn hình chính
- Check box Tăng dần âm lượng báo thức.
- Check box Ngăn tắt nguồn điện thoại khi báo thức reo.
- 2 Radio button chọn cách hiển thị giờ (24 or 12)

3. Màn hình cài đặt của từng báo thức:

- Tên người làm: Nguyễn Anh Nhật
- Hình:



- Chức năng của màn hình này: Thiết lập các cài đặt của của báo thức như chọn giờ, nhạc chuông, cách tắt báo thức, đặt tên báo thức,...


- Các Button:


+ Hủy: Hủy cài đặt/tạo báo thức

+ Xóa: Xóa báo thức đang chọn

+ OK: Lưu thông tin báo thức vào database và trở về màn hình chính

- Các ImageButton:

+ Play music : Nghe thử nhạc chuông báo thức.

+ Âm lượng : Tắt/mở âm lượng chuông báo thức.

- Time Picker: Chọn giờ báo thức.

- Text View: 

+ “+10M”: Cộng thêm 10p vào Time Picker

+ “-10M”: Trừ đi 10p vào Time Picker

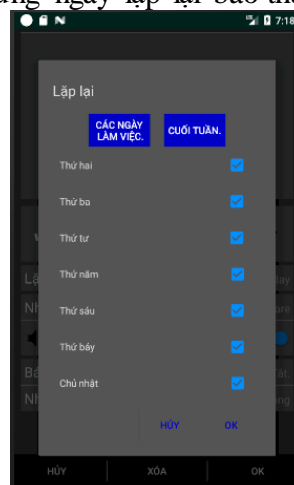
+ “+1H”: Cộng 1 giờ vào Time Picker

+ “-1H”: Trừ 1 giờ vào Time Picker

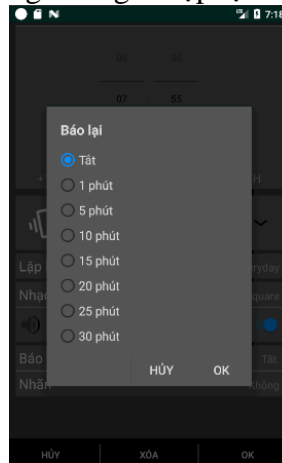
- LinearLayout:

+ Cách tắt báo thức: Chuyển đến màn hình chọn cách tắt báo thức.

+ Lặp lại: Hiện thị fragment để chọn những ngày lặp lại báo thức.





- + Nhạc chuông: Chuyển tới màn hình chọn nhạc chuông cho báo thức.
- + Báo lại: Hiển thị fragment để chọn khoảng thời gian lặp lại báo thức.



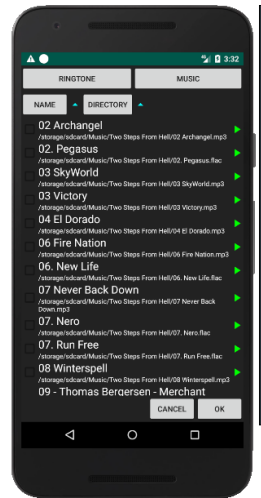
- + Nhãn báo thức: đặt tên cho báo thức.



- Seekbar : Điều chỉnh âm lượng của báo thức.
- Switch : Chọn rung hoặc không rung khi báo thức reo.

4. Màn hình chọn nhạc chuông:

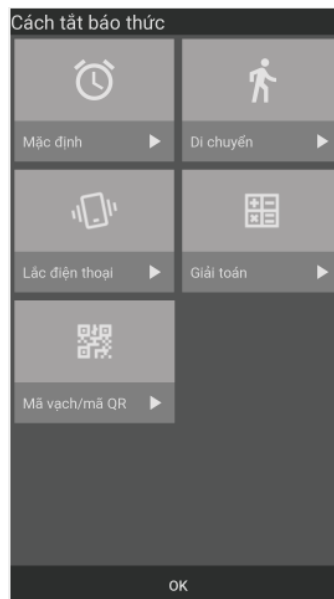
- Tên người làm: Đào Phi Lạc
- Hình:




- Chức năng của màn hình này: Chọn nhạc chuông cho báo thức.
- Các button:
 - + Ring tone: Liệt kê các nhạc chuông mặc định có trong điện thoại.
 - + Music: Liệt kê các bản nhạc của người dùng.
 - + Tên: Sắp xếp nhạc chuông theo tên tăng or giảm dần.
 - + Đường dẫn: Sắp xếp theo đường dẫn
 - + Cancel: Thôi.
 - + OK: Xác nhận nhạc chuông đã chọn và trở về màn hình cài đặt báo thức.
 - + Nghe thử nhạc chuông.
- Check box: chọn bản nhạc chuông tương ứng

5. Màn hình chọn cách tắt báo thức:

- Tên người làm: Nguyễn Anh Nhật
- Hình:

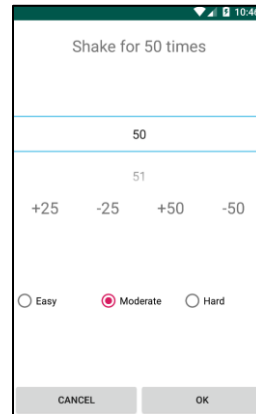


- Chức năng của màn hình này: Chọn thử thách để tắt được báo thức khireo.
- Các Button:

- + Xem trước : Xem trước giao diện tương ứng lúc báo thức reo.
- + OK: Xác nhận cách tắt báo thức đã chọn và chuyển về màn hình cài đặt báo thức.
- LinearLayout: Hiện app đã có 4 cách tắt báo thức:
 - + Mặc định: Chọn cách tắt báo thức mặc định.
 - + Lắc: Chọn cách tắt báo thức là lắc, chuyển tới màn hình cài đặt chế độ lắc.
 - + Làm toán: Chọn cách tắt báo thức là làm toán, chuyển tới màn hình cài đặt chế độ làm toán.
 - + Di chuyển: Chọn cách tắt báo thức là di chuyển, chuyển tới màn hình cài đặt chế độ di chuyển.

6. Màn hình cài đặt chế độ lắc:

- Tên người làm: Nguyễn Đào Hồng Nhung
- Hình:



- Chức năng của màn hình này:
- Các button:
 - + OK: Xác nhận tùy chọn đã thiết lập và chuyển về màn hình chọn cách tắt báo thức
 - + Cancel: chuyển về màn hình chọn cách tắt báo thức
- Number Picker: Chọn số lần lắc.
- Text View:
 - + Cộng 25 đơn vị vào Number Picker
 - + Trừ 25 đơn vị vào Number Picker
 - + Cộng 50 đơn vị vào Number Picker
 - + Trừ 50 đơn vị vào Number Picker
- Radio button: Chọn mức độ khó khi lắc
 - + Dễ
 - + Vừa
 - + Khó

7. Màn hình thử thách lắc:

- Tên người làm: Đào Phi Lạc
- Hình:



- Chức năng của màn hình này: Hiện lên thử thách và buộc người dùng phải lắc điện thoại đủ số lần thì báo thức mới tắt

- Các button:

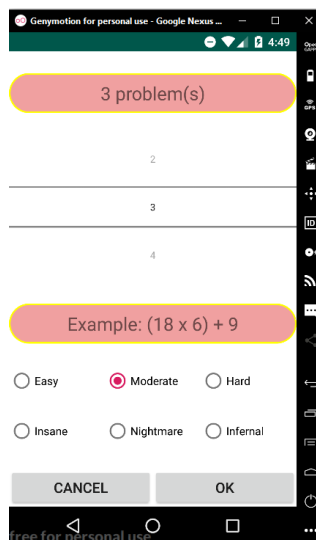
+ Im lặng: Ngừng reo báo thức trong thời gian đã thiết đặt trong cài đặt.

+ Bỏ cuộc: Chuyển sang màn hình bấm nút 500 lần để tắt báo thức.

8. Màn hình cài đặt chế độ làm toán:

- Tên người làm: Trần Thị Thanh

- Hình:



- Chức năng của màn hình này:

- Các button:

+ OK: Xác nhận tùy chọn đã thiết lập và chuyển về màn hình chọn cách tắt báo thức

- + Cancel: chuyển về màn hình chọn cách tắt báo thức
- + Easy, Moderate, Hard, insane, nightmare, infernal: Chọn mức độ khó của phép toán.
- TextView:
 - + Number_of_problems: Hiển thị số phép toán người chơi muốn giải khi báo thức
 - + Example: Hiển thị mẫu phép toán với từng mức độ khó mà người chơi chọn.

9. Màn hình thử thách làm toán:

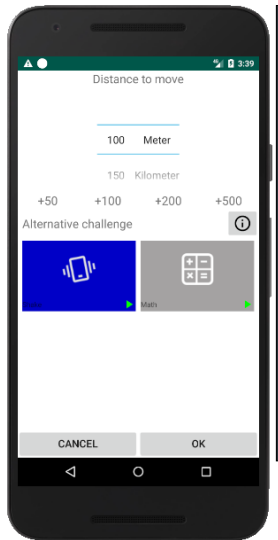
- Tên người làm: Trần Thị Thanh
- Hình:



- Chức năng của màn hình này:
 - + Hiển thị phép toán để người dùng giải khi báo thức bắt đầu reo (Nếu người chơi lựa chọn thử thách là giải toán).
 - + Số phép toán được người chơi thiết đặt từ trước.
 - + Khi người chơi nhập xong kết quả và bấm OK
 - + Các phép toán được random theo các mức độ: Easy, Moderate, Hard, insane, nightmare, infernal
 - + Khi báo thức ngừng thì chuyển về màn hình chính.
- Các button:
 - + Im lặng: Ngừng reo báo thức trong thời gian đã thiết đặt trong cài đặt.
 - + Bỏ cuộc: Chuyển sang màn hình bấm nút 500 lần để tắt báo thức.
 - + Bàn phím số để người dùng nhập kết quả
 - + OK: Xác nhận kết quả
 - + Xóa: Xóa kết quả.
- TextView:
 - + TextView hiển thị phép toán
 - + TextView hiển thị kết quả do người dùng nhập vào

10. Màn hình cài đặt thử thách di chuyển

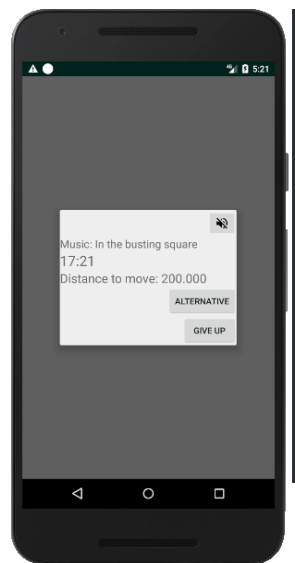
- Tên người làm: Đào Phi Lạc
- Hình:



- Chức năng: thiết lập thông tin của thử thách di chuyển
- Các controls:
 - + DistancePicker: chọn khoảng cách cần phải di chuyển
 - + TextView +50: cộng thêm 50 mét vào DistancePicker
 - + TextView +100: cộng thêm 100 mét vào DistancePicker
 - + TextView +200: cộng thêm 200 mét vào DistancePicker
 - + TextView +500: cộng thêm 500 mét vào DistancePicker
 - + ConstraintLayout Shake: chuyển sang màn hình cài đặt thử thách lắc thay thế
 - + ConstraintLayout Math: chuyển sang màn hình cài đặt thử thách toán thay thế
 - + Cancel: hủy cài đặt thử thách di chuyển
 - + OK: đồng ý cài đặt và ghi thông tin vào database

11. Màn hình thử thách di chuyển

- Tên người làm: Đào Phi Lạc
- Hình



- Chức năng: Hiện lên thử thách và buộc người dùng phải di chuyển đủ số mét thì báo thức mới tắt
- Các controls:
 - + Im lặng: im lặng báo thức
 - + Give up: chuyển đến màn hình bỏ cuộc, buộc người dùng nhấn một nút 500 lần để tắt báo thức
 - + Alternative: chuyển đến màn hình thử thách thay thế, trong trường hợp người dùng không thể di chuyển được.

II. SDS (Software Design Specification)


1. Màn hình chính:


- Tên người làm: Mạc Đỗ Xuân Tính


- Xử lý:



+ Bấm vào Item báo thức : Hiển thị màn hình cài đặt báo thức ở chế độ SỬA.

+ Bấm vào nút  : Hiển thị màn hình cài đặt báo thức ở chế độ THÊM MỚI.

+ Bấm vào nút  : Chuyển sang trang màn hình các báo thức (khi đang ở màn hình Cài đặt).

+ Bấm vào nút  : Chuyển sang trang màn hình Cài đặt cho ứng dụng (khi đang ở màn hình các báo thức)

+ Bấm vào nút  : Bật/Tắt báo thức.

2. Màn hình cài đặt của app:

- Tên người làm: Nguyễn Đào Hồng Nhung

- Xử lý:

+ Khi người dùng bấm vào button thời gian im lặng của báo thức thì một Fragment hiện lên cho phép người dùng lựa chọn thời gian tắt tiếng chuông báo một khi báo thức đang diễn ra. Với thời gian nhỏ nhất là 30 giây, lớn nhất là 300 giây

+ Khi người dùng bấm vào button số lần im lặng của báo thức thì một Fragment hiện lên cho phép người dùng lựa chọn số lần im lặng khi một báo thức đang diễn ra. Với giá trị nhỏ nhất là 0 và lớn nhất là 5

+ Khi người dùng bấm vào button tự động hủy báo thức thì một Fragment hiện lên cho phép người dùng lựa chọn thời gian tự động hủy báo thức. Với thời gian nhỏ nhất là 10 phút và lớn nhất là 60 phút.

+ Người dùng có 2 check box Tăng dần âm lượng và Ngăn chặn tắt điện thoại

+ Người dùng có 2 radio button cài đặt định dạng giờ của báo thức 24 giờ hoặc 12 giờ

3. Màn hình cài đặt của từng báo thức:

- Tên người làm: Nguyễn Anh Nhật

- Xử lý:

+ Khi fragment hiện lên, app sẽ kiểm tra xem đang ở chế độ sửa báo thức, hay tạo mới. Nếu là sửa, các dữ liệu của báo thức cũ sẽ hiện lên cho từng mục. Nếu tạo mới thì các giá trị là mặc định.

+ Người dùng cài đặt xong nhấn OK, các dữ liệu thay đổi sẽ lưu vào database.

- + Khi nhấn Hủy: Quay về màn hình chính mà không lưu các thay đổi đã thực hiện.
- + Khi nhấn Xóa(chỉ xóa được khi đang ở chế độ sửa báo thức): Dữ liệu của báo thức sẽ xóa khỏi database, và quay về màn hình chính.

4. Màn hình chọn nhạc chuông:

- Tên người làm: Đào Phi Lạc
- Xử lý:
 - + Bấm vào nút RINGTONE thì liệt kê ra danh sách nhạc chuông mặc định của điện thoại.
 - + Bấm vào nút MUSIC thì liệt kê ra danh sách những bản nhạc hiện có trong máy. Các bản nhạc là các file có đuôi mp3 m4a wav ogg flac
 - + Bấm vào NAME thì sắp xếp các bản nhạc theo tên. Nếu đang sắp tăng dần thì sắp giảm dần, nếu đang sắp giảm dần thì sắp tăng dần.
 - + Bấm vào DIRECTORY thì sắp xếp các bản nhạc theo đường dẫn. Nếu đang sắp tăng dần thì sắp giảm dần, nếu đang sắp giảm dần thì sắp tăng dần.
 - + Chọn vào CheckBox thì ghi nhớ bản nhạc tương ứng để khi người dùng chọn OK thì lưu đường dẫn của bản nhạc đó vào thông tin của báo thức.
 - + Bấm vào nút Play thì chơi bản nhạc tương ứng.
 - + Bấm Cancel thì quay trở về màn hình trước đó.
 - + Bấm OK thì kiểm tra xem có đang có bản nhạc nào được chọn hay không, nếu có thì lưu vào thông tin của báo thức tương ứng rồi lưu thay đổi vào database.

5. Màn hình chọn cách tắt báo thức:

- Tên người làm: Nguyễn Anh Nhật
- Xử lý:
 - + Khi người dùng nhấn vào các layout, thì layout đó sẽ chuyển sang màu xanh. Ở cách lắc và làm toán sẽ hiển thị form để cài đặt cái tùy chọn.
 - + Khi bấm vào ImageButton play, thì các màn hình xem trước hiển thị lên.
 - + Khi bấm OK, xác nhận cách tắt báo thức là cách đã chọn.

6. Màn hình cài đặt chế độ lắc:

- Tên người làm: Nguyễn Đào Hồng Nhung
- Xử lý:
 - + Number Picker: Số lần lắc điện thoại
 - + Khi người dùng click vào text view +25 thì Number Picker sẽ cộng 25 đơn vị và hiển thị Number Picker tương ứng trên text view Shar for...times
 - + Khi người dùng click vào text view -25 thì Number Picker sẽ trừ 25 đơn vị và hiển thị Number Picker tương ứng trên text view Shar for...times
 - + Khi người dùng click vào text view +50 thì Number Picker sẽ cộng 50 đơn vị và hiển thị Number Picker tương ứng trên text view Shar for...times
 - + Khi người dùng click vào text view -50 thì Number Picker sẽ trừ 50 đơn vị và hiển thị Number Picker tương ứng trên text view Shar for...times
 - + Nếu như Number Picker có giá trị 50 thì người dùng chỉ có thể tăng thêm số lần lắc chứ không thể giảm
 - + Người dùng có 3 sự lựa chọn mức độ lắc tương ứng 3 radio button: dễ, vừa, khó

7. Màn hình thử thách lắc:

- Tên người làm: Đào Phi Lạc
- Xử lý:
 - + Bấm vào nút Mute: tạm thời giữ yên lặng, không phát nhạc chuông trong một khoảng thời gian được đặt trước trong phần cài đặt của app. Trong phần cài đặt của app cũng có quy định số lần tối đa được phép bấm nút này, bấm quá số lần sẽ không có tác dụng
 - + Bấm vào nút Give Up: chuyển đến màn hình bỏ cuộc, buộc người dùng phải bấm một nút 500 lần để tắt báo thức.
 - + Bấm vào nút Alternative: chuyển đến màn hình thử thách thay thế trong trường hợp người dùng không thể di chuyển được.

8. Màn hình cài đặt chế độ làm toán:

- Tên người làm: Trần Thị Thanh
- Xử lý:
 - + Người dùng chọn số phép tính để tính khi báo thức kêu thông qua NumberPicker.
 - + Mức độ khó của phép toán là các radio button, khi người dùng chọn mức độ khó nào sẽ được lưu lại và gửi qua màn hình thử thách làm toán.
 - + Khi người chơi chọn mức độ khó thì textView example sẽ hiển thị phép toán ví dụ của mức độ.(Phép toán được random khi xử lý màn hình thử thách làm toán)

9. Màn hình thử thách làm toán:

- Tên người làm: Trần Thị Thanh
- Xử lý:
 - + Khi người dùng bấm số trên bàn phím số, các số được gán và một TextView Kết quả để hiển thị.
 - + Khi bấm Xóa, kết quả hiển thị sẽ xóa cho tới hết số trên TextView.
 - + Khi bấm OK, kiểm tra kết quả người dùng nhập vào với kết quả đã tính được khi random số theo phép toán định sẵn.
- Nếu kết quả đúng, số phép toán làm đúng bằng với số phép toán người chơi đã cài đặt trước, thì tắt báo thức
- Nếu kết quả đúng, số phép toán đúng chưa bằng số phép toán người chơi cài đặt trước thì sẽ hiển thị tính tiếp theo.
- Nếu kết quả sai, không chuyển phép tính.

10. Màn hình cài đặt thử thách di chuyển

- Tên người làm: Đào Phi lạc
- Xử lý:
 - + Cuộn picker để chọn số mét cho thử thách.
 - + Có thể bấm vào các TextView +50 +100 +200 +500 để cộng thêm số mét vào picker.
 - + Picker có bước nhảy là 50, giá trị nhỏ nhất là 100. Nếu người dùng chọn đơn vị là Kilomet thì picker sẽ chuyển sang bước nhảy là 1, giá trị nhỏ nhất là 1, đồng thời ẩn đi các TextView
 - + Thử thách lắc và toán thay thế sẽ chuyển đến màn hình cài đặt lắc hoặc toán để thiết lập một thử thách thay thế khác trong trường hợp người dùng không thể di chuyển được trong tương lai vì một lý

do nào đó

- + Bấm Cancel để hủy mọi cài đặt và trở về màn hình trước đó
- + Bấm OK thì chương trình kiểm tra xem số mét có lớn hơn 3000 hay không, nếu có thì hỏi lại người dùng là có chắc chắn với cài đặt này hay chưa

11. Màn hình thử thách di chuyển:

- Tên người làm: Đào Phi Lạc

- Xử lý:

- + Bấm vào nút Mute: tạm thời giữ yên lặng, không phát nhạc chuông trong một khoảng thời gian được đặt trước trong phần cài đặt của app. Trong phần cài đặt của app cũng có quy định số lần tối đa được phép bấm nút này, bấm quá số lần sẽ không có tác dụng
- + Bấm vào nút Give Up: chuyển đến màn hình bỏ cuộc, buộc người dùng phải bấm một nút 500 lần để tắt báo thức.
- + Bấm vào nút Alternative: chuyển đến màn hình thử thách thay thế trong trường hợp người dùng không thể di chuyển được.

Chương 3: Cài đặt và kiểm thử

1. Màn hình chính:

- Tên người làm: Mạc Đỗ Xuân Tính

a. Màn hình chính: MainActivity

- Chứa các nút thêm báo thức, nút chuyển sang màn hình cài đặt ứng dụng.
- Hỏi quyền truy cập bộ nhớ ngoài sau khi ứng dụng được mở lần đầu sau khi cài đặt:

```
PermissionInquirer permissionInquirer = new PermissionInquirer( context: this);
permissionInquirer.askPermission(Manifest.permission.READ_EXTERNAL_STORAGE, permissionCode: 1);

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch(requestCode){
        case 1:
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            } else {
                Toast.makeText( context: this, text: "Read external storage permission has not been granted. Terminate!", Toast.LENGTH_LONG).show();
                finish();
            }
            break;
    }
}
```

- + Nếu người dùng không cho phép, ứng dụng sẽ hiện thông báo và tắt:

```
Resources resources = getResources();
int resId = R.raw.in_the_busting_square;
Music.defaultRingtoneUrl = ContentResolver.SCHEME_ANDROID_RESOURCE + "://" + resources.getResourcePackageName(resId) + '/' + resources.getResourceTypeName(resId) + '/'

appSettingFragment = new AppSettingFragment();
appSettingFragment.loadAppSetting( context: this);
appSettingFragment.setEnterTransition(new Slide(Gravity.END));
appSettingFragment.setExitTransition(new Slide(Gravity.END));

upcomingAlarmFragment = new UpcomingAlarmFragment();
fragmentManager = getSupportFragmentManager();
fragmentManager.beginTransaction().add(R.id.main_fragment_container, upcomingAlarmFragment).commit();
NotificationService.update( context: this);
```

- Nạp vào âm thanh báo thức mặc định, và các Fragment màn hình danh sách báo thức, màn hình cài

đặt ứng dụng.

- Xử lý nút Báo thức và nút Cài đặt:

```
private void setEvent() {
    buttonAlarm.setOnClickListener(onClick(v) → {
        if(appSettingFragmentIsAdded){
            fragmentManager.popBackStack();
        }
    });
    buttonSetting.setOnClickListener((v) → {
        if(!appSettingFragmentIsAdded){
            appSettingFragmentIsAdded = true;
            fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.add(R.id.main_fragment_container, appSettingFragment);
            fragmentTransaction.addToBackStack(null);
            fragmentTransaction.commit();
        }
    });
}
```

b. Màn hình các báo thức: UpcomingAlarmFragment.

- Nạp vào màn hình Cài đặt của báo thức, nạp vào danh sách các báo thức từ Database hiển thị lên RecyclerView thông qua Adapter.

```
public void onAttach(final Context context) {
    super.onAttach(context);
    mainActivity = (MainActivity) context;
    databaseHandler = new DatabaseHandler(getContext());
    settingAlarmFragment = new SettingAlarmFragment();
    / settingAlarmFragment.setEnterTransition(R.anim.anim_button_add_alarm_press);
    / settingAlarmFragment.setExitTransition(new Slide(Gravity.END));
    listAlarm = databaseHandler.getAllAlarm();
    Collections.sort(listAlarm);
    alarmAdapter = new AlarmAdapter(getContext(), upcomingAlarmFragment: this, listAlarm);
    fragmentManager = getFragmentManager();
    mainActivity.appSettingFragment.setOnHourModeChangeListener((hourMode) → {
        alarmAdapter.notifyDataSetChanged();
        NotificationService.changeHourMode(context);
    });
}
```

```
recyclerViewListAlarm.setAdapter(alarmAdapter);
recyclerViewListAlarm.setLayoutManager(new LinearLayoutManager(getContext()));
```

- Hiển thị thời gian còn lại từ thời điểm hiện tại của hệ thống đến báo thức được bật sắp tới:

```

private void setTimeRemaining() {
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        Calendar calendarAlarm = Calendar.getInstance();
        Calendar calendarNow = Calendar.getInstance();
        long timeDelta;
        long days, hours, minutes;
        String time = "";

        @Override
        public void run() {
            Alarm alarm = databaseHandler.getTheNearestAlarm();
            if (alarm == null) {
                tvTimeRemaining.setText("Off");
            } else {
                calendarAlarm.set( year: 1970, month: 1, date: 1, hourOfDay: 1, minute: 1, second: 1);
                calendarNow.set( year: 1970, month: 1, date: 1, hourOfDay: 1, minute: 1, second: 1);
                //Log.d("MILLI", "TotalMilli: " + calendarNow.getTimeInMillis() + ", " + calendarAlarm.getTimeInMillis());
                calendarNow.set(Calendar.DAY_OF_WEEK, Calendar.getInstance().get(Calendar.DAY_OF_WEEK));
                calendarNow.set(Calendar.HOUR_OF_DAY, Calendar.getInstance().get(Calendar.HOUR_OF_DAY));
                calendarNow.set(Calendar.MINUTE, Calendar.getInstance().get(Calendar.MINUTE));
                calendarNow.set(Calendar.SECOND, Calendar.getInstance().get(Calendar.SECOND));
                //Log.d("WEEKMONTH", ""+calendarNow.get(Calendar.WEEK_OF_MONTH)+", "+calendarAlarm.get(Calendar.WEEK_OF_MONTH));
                calendarAlarm.set(Calendar.DAY_OF_WEEK, alarm.getDayOfWeek());
                if (calendarAlarm.get(Calendar.DAY_OF_WEEK) < calendarNow.get(Calendar.DAY_OF_WEEK)) {
                    calendarAlarm.set(Calendar.WEEK_OF_MONTH, calendarNow.get(Calendar.WEEK_OF_MONTH) + 1);
                }
                //Log.d("WEEKMONTH2", "" + calendarNow.get(Calendar.WEEK_OF_MONTH) + ", " + calendarAlarm.get(Calendar.WEEK_OF_MONTH));
                calendarAlarm.set(Calendar.HOUR_OF_DAY, alarm.getHour());
                calendarAlarm.set(Calendar.MINUTE, alarm.getMinute());
                calendarAlarm.set(Calendar.SECOND, 0);
                timeDelta = Math.abs(calendarAlarm.getTimeInMillis() - calendarNow.getTimeInMillis());

                days = TimeUnit.MILLISECONDS.toDays(timeDelta);
                hours = TimeUnit.MILLISECONDS.toHours( duration: timeDelta - days * 24 * 60 * 60 * 1000);
                minutes = TimeUnit.MILLISECONDS.toMinutes( duration: timeDelta - days * 24 * 60 * 60 * 1000 - hours * 60 * 60 * 1000);

                if (days > 0) {
                    time = days + " days ";
                }
                if (hours > 0) {
                    time = time + hours + " hours ";
                }
                if (minutes > 0) {
                    time = time + minutes + " minutes ";
                }
                if (timeDelta < 60000) {
                    time = "Less than 1 minute ";
                }
                tvTimeRemaining.setText(time + "remaining");
                time = "";
                timeDelta = 0;
            }
            handler.postDelayed( r: this, delayMillis: 1000);
        }
    }, delayMillis: 1000);
}

```

- Nhận các thiết lập đã được cài đặt ở màn hình Cài đặt chi tiết cho báo thức, thực hiện cập nhật trong Database (Thêm/Xóa/Sửa), cập nhật RecyclerView danh sách các báo thức,...

```

public void addAlarm(Alarm alarm) {
    alarm.setChallengeType(DEFAULT);
    this.databaseHandler.insertAlarm(alarm);
    alarm.setIdAlarm(databaseHandler.getRecentAddedAlarm().getIdAlarm());
    listAlarm.add(alarm);
    Collections.sort(listAlarm);
    alarmAdapter.notifyDataSetChanged();
    NotificationService.update(getContext());
}

```

```

public void editAlarm(Alarm alarm) {
    if (alarm.getChallengeType() != DEFAULT) {
        this.databaseHandler.deleteChallengeDetail(alarm.getIdAlarm(), alarm.getChallengeType())
    }
    alarm.setChallengeType(DEFAULT);
    this.databaseHandler.updateAlarm(alarm);
    for (int i = 0; i < listAlarm.size(); i++) {
        if (listAlarm.get(i).getIdAlarm() == alarm.getIdAlarm()) {
            listAlarm.set(i, alarm);
            Collections.sort(listAlarm);
            alarmAdapter.notifyDataSetChanged();
            break;
        }
    }
    NotificationService.update(getContext());
}

public void deleteAlarm(int idAlarm) {
    databaseHandler.deleteAlarm(idAlarm);
    for (int i = 0; i < listAlarm.size(); i++) {
        if (listAlarm.get(i).getIdAlarm() == idAlarm) {
            listAlarm.remove(i);
            alarmAdapter.notifyItemRemoved(i);
            NotificationService.update(getContext());
            break;
        }
    }
}
}

```

c. AlarmAdapter:

- Dữ liệu được đổ ra RecyclerView để hiển thị thông qua AlarmAdapter:

```
public class AlarmAdapter extends RecyclerView.Adapter<AlarmAdapter.AlarmViewHolder> {
    private Context context;
    private UpcomingAlarmFragment upcomingAlarmFragment;
    private List<Alarm> listAlarm;
    private SettingAlarmFragment settingAlarmFragment;
    private FragmentManager fragmentManager;
    private HashMap<ConstraintLayout, Alarm> mapViewAlarm;

    public AlarmAdapter(Context context, UpcomingAlarmFragment upcomingAlarmFragment, List<Alarm> listAlarm) {
        this.context = context;
        this.upcomingAlarmFragment = upcomingAlarmFragment;
        this.listAlarm = listAlarm;
        this.settingAlarmFragment = new SettingAlarmFragment();
        this.fragmentManager = ((FragmentActivity) context).getSupportFragmentManager();
        this.mapViewAlarm = new HashMap<>();
    }

    @Override
    public int getItemCount() { return this.listAlarm.size(); }

    @{...}
    public AlarmViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {...}

    @Override
    public void onBindViewHolder(@NonNull final AlarmViewHolder alarmViewHolder, int i) {...}

    class AlarmViewHolder extends RecyclerView.ViewHolder {
        private ConstraintLayout constraintLayoutParent;
        private SwitchCompat swcEnable;
        private TextView tvHour;
        private TextView tvMinute;
        private TextView tvRepeatDay;
        private TextView tvAMPM;
        private ImageButton btnAlarmType;
    }
}
```

+ Thiết lập kiểu hiển thị giờ (12h hoặc 24h):

```
int hour = alarm.getHour();
if (AppSettingFragment.hourMode == HOUR_MODE_24) {
    tvAMPM.setText("");
    if (hour < 10) {
        tvHour.setText("0" + hour);
    } else {
        tvHour.setText(String.valueOf(hour));
    }
} else {
    if (hour < 12) {
        tvAMPM.setText("AM");
        if (hour == 0) {
            tvHour.setText("12");
        } else if (hour < 10) {
            tvHour.setText("0" + hour);
        } else tvHour.setText("" + hour);
    } else { //hour >=12
        tvAMPM.setText("PM");
        if (hour == 12) tvHour.setText("" + hour);
        else {
            if ((hour - 12) < 10) {
                tvHour.setText("0" + (hour - 12));
            } else tvHour.setText("" + (hour - 12));
        }
    }
}
}
```

+ Thực hiện mở màn hình Cài đặt chi tiết cho báo thức khi nhấn vào Item báo thức:

```

constraintLayoutParent.setOnClickListener((v) -> {
    Alarm checkedAlarm = mapViewAlarm.get(constraintLayoutParent);
    checkedAlarm.validateRingtoneUrl(context);
    settingAlarmFragment.configure(upcomingAlarmFragment, checkedAlarm);
    fragmentManager.beginTransaction()
        .add(R.id.full_screen_fragment_container, settingAlarmFragment)
        .addToBackStack(null).commit();
});
btnAlarmType.setOnClickListener((v) -> {
    Alarm checkedAlarm = mapViewAlarm.get(constraintLayoutParent);
    checkedAlarm.validateRingtoneUrl(context);
    settingAlarmFragment.configure(upcomingAlarmFragment, checkedAlarm);
    fragmentManager.beginTransaction()
        .add(R.id.full_screen_fragment_container, settingAlarmFragment)
        .addToBackStack(null).commit();
});

```

+ Cập nhật lại vị trí các báo thức hiển thị trên RecyclerView khi Bật/Tắt Báo thức:

```

swcEnable.setOnClickListener((v) -> {
    alarm.setEnabled(swcEnable.isChecked());
    upcomingAlarmFragment.updateAlarmEnable(alarm);
    listAlarm.set(alarmViewHolder.getAdapterPosition(), alarm);
    Collections.sort(listAlarm);
    AlarmAdapter.this.notifyDataSetChanged();
});

```

2. Màn hình cài đặt của app:

- Tên người làm: Nguyễn Đào Hồng Nhung

a. Thời gian im lặng của báo thức

Khi click vào button đứng ngang hàng với text “Báo thức có thể im lặng trong: “ thì một Fragment hiện ra để người dùng lựa chọn

```

btnMute.setOnClickListener((v) -> {
    muteAlarmInDialogFragment.show(getFragmentManager(), tag: null);
});

```

b. Số lần im lặng của báo thức

Khi click vào button đứng ngang hàng với text “Có thể im lặng báo thức: “ thì một Fragment hiện ra để người dùng lựa chọn

```

btnCanMute.setOnClickListener((v) -> {
    canMuteAlarmForDialogFragment.show(getFragmentManager(), tag: null);
});

```

c. Tự động tắt báo thức

Khi click vào button đứng ngang hàng với text “Tự động hủy báo thức sau: “ thì một Fragment hiện ra để người dùng lựa chọn

```
btnDismiss.setOnClickListener((v) → {
    autoDismissAfterDialogFragment.show(getFragmentManager(), tag: null);
});
```

d. Nút Back: quay lại màn hình chính

```
this.imageButtonBack.setOnClickListener((v) → {
    getFragmentManager().beginTransaction().remove(AppSettingFragment.this).commit();
});
```

e. Check box Tăng dần âm lượng báo thức

```
this.checkBoxGraduallyIncreaseVolume.setOnCheckedChangeListener((buttonView, isChecked) → {
    graduallyIncreaseVolume = isChecked;
});
```

f. Check box Ngăn tắt nguồn điện thoại khi báo thức reo

```
this.checkBoxPreventTurnOffPhone.setOnCheckedChangeListener((buttonView, isChecked) → {
    preventTurnOffPhone = isChecked;
});
```

g. Radio button chọn cách hiển thị giờ (24 or 12)

```
this.radioButton24Hour.setOnCheckedChangeListener((buttonView, isChecked) → {
    if(isChecked){
        hourMode = HOUR_MODE_24;
        if(onHourModeChangeListener != null){
            onHourModeChangeListener.onHourModeChange(hourMode);
        }
    }
});
this.radioButton12Hour.setOnCheckedChangeListener((buttonView, isChecked) → {
    if(isChecked){
        hourMode = HOUR_MODE_12;
        if(onHourModeChangeListener != null){
            onHourModeChangeListener.onHourModeChange(hourMode);
        }
    }
});
```

h. Hiệu ứng animation: fade in và blink

```
layoutSetting = view.findViewById(R.id.layoutSetting);

animationFadein = AnimationUtils.loadAnimation(getContext(), R.anim.fade_in);
animationFadein.setAnimationListener(this);
layoutSetting.startAnimation(animationFadein);

animBlink = AnimationUtils.loadAnimation(getContext(), R.anim.blink);
animBlink.setAnimationListener(this);
imageButtonBack.startAnimation(animBlink);
```

3. Màn hình cài đặt của từng báo thức:

- Tên người làm: Nguyễn Anh Nhật
- Xử lý:

+ Nhấn OK xác nhận cài đặt:

```
btnOk.setOnClickListener((view) -> {
    outputAgain = textViewAgain.getText().toString();
    label = textViewLabel.getText().toString();

    if (Build.VERSION.SDK_INT >= 23) {
        hour = timePicker.getHour();
        minute = timePicker.getMinute();
    }
    else {
        hour = timePicker.getCurrentHour();
        minute = timePicker.getCurrentMinute();
    }
    timePicker.getCurrentHour();
    volume = seekBar.getProgress();
    if (aSwitch.isChecked()) vibrate = true;
    else vibrate = false;

    alarm.setHour(getHour(timePicker));
    alarm.setMinute(getMinute(timePicker));
    alarm.setVibrate(aSwitch.isChecked());
    alarm.setSnoozeTime(snoozeTime);
    alarm.setVolume(seekBar.getProgress());
    if (settingAlarmMode == SettingAlarmMode.EDIT) {
        editAlarm();
    } else if (settingAlarmMode == SettingAlarmMode.ADD_NEW) {
        addAlarm();
    }
    fragmentManager.popBackStack();
});
```

+ Xóa và Hủy cài đặt:

```
btnDelete.setOnClickListener((v) -> {
    if (settingAlarmMode == SettingAlarmMode.EDIT) {
        upcomingAlarmFragment.deleteAlarm(alarm.getIdAlarm());
        fragmentManager.popBackStack();
    }
});

btnCancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        fragmentManager.popBackStack();
    }
});
```

4. Màn hình chọn nhạc chuông:

- Tên người làm: Đào Phi Lạc

```
buttonRingtone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (chooseMusicType != ChooseMusicType.RINGTONE) {
            chooseMusicType = ChooseMusicType.RINGTONE;
            makeListRingtone();
        }
    }
});

buttonMusic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (chooseMusicType != ChooseMusicType.MUSIC) {
            chooseMusicType = ChooseMusicType.MUSIC;
            makeListMusic();
        }
    }
});
```

+ Nút Ringtone sẽ tạo list nhạc chuông mặc định rồi gán cho RecyclerView

+ Nút Music sẽ tạo list nhạc chuông bằng cách tìm các file có đuôi mp3 m4a wav ogg flac trong máy rồi gán cho RecyclerView

```
buttonSortByName.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isSortingBy != SortBy.NAME) {
            isSortingBy = SortBy.NAME;
            isWithOrder = Order.ASC;
            Collections.sort(listMusic, new Music.NameComparator());
            imageViewSortByName.setImageDrawable(getContext().getDrawable(R.drawable.ic_arrow_drop_up));
        }
        else if (isWithOrder == Order.ASC) {
            isWithOrder = Order.DESC;
            Collections.sort(listMusic, new Music.NameComparator());
            Collections.reverse(listMusic);
            imageViewSortByName.setImageDrawable(getContext().getDrawable(R.drawable.ic_arrow_drop_down));
        }
        else if (isWithOrder == Order.DESC) {
            isWithOrder = Order.ASC;
            Collections.sort(listMusic, new Music.NameComparator());
            imageViewSortByName.setImageDrawable(getContext().getDrawable(R.drawable.ic_arrow_drop_up));
        }
        musicAdapter.notifyDataSetChanged();
    }
});
buttonSortByUrl.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isSortingBy != SortBy.URL) {...}
        else if (isWithOrder == Order.ASC) {...}
        else if (isWithOrder == Order.DESC) {...}
        musicAdapter.notifyDataSetChanged();
    }
});
```

+ Nút SortByName để sắp xếp danh sách các bản nhạc trong RecyclerView theo tên. Nếu như hiện tại đang sắp tăng dần thì sắp giảm dần, nếu đang sắp giảm dần thì sắp tăng dần. Nếu đang sắp theo đường dẫn thì sắp tăng dần.

+ Nút SortByUrl để sắp xếp danh sách các bản nhạc trong RecyclerView theo đường dẫn. Nếu như hiện tại đang sắp tăng dần thì sắp giảm dần, nếu đang sắp giảm dần thì sắp tăng dần. Nếu đang sắp theo tên thì sắp tăng dần.

5. Màn hình chọn cách tắt báo thức:

- Tên người làm: Nguyễn Anh Nhật

- Các xử lý:

+ Xem trước các màn hình khi báo thức reo, khi nhấn vào cái ImageButton:


```

imageButtonDefault.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), TypePlayDefaultActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
        startActivity(intent);
    }
});
imageButtonMoving.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), TypePlayCameraActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
        startActivity(intent);
    }
});
imageButtonShake.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), TypePlayShakeActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
        startActivity(intent);
    }
});
imageButtonMath.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), TypePlayMathActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
        startActivity(intent);
    }
});

```

+ Chọn cách tắt báo thức:

```

linearLayoutDefault.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        typeFragmentListener.getDefaultChallenge();
        updateChallengeLayoutColor(ChallengeType.DEFAULT);
    }
});
linearLayoutMath.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mathConfigurationFragment.setAlarm(alarm);
        fragmentManager.beginTransaction()
            .add(R.id.full_screen_fragment_container, mathConfigurationFragment)
            .addToBackStack(null)
            .commit();
        mathConfigurationFragment.setMathConfigurationFragmentListener(new MathConfigurationFragment
            .MathConfigurationFragmentListener() {
                @Override
                public void onMathConfigurationSetup(MathDetail mathDetail) {
                    typeFragmentListener.getMathChallenge(mathDetail);
                    updateChallengeLayoutColor(ChallengeType.MATH);
                }
            });
    }
});
linearLayoutShake.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        shakeConfigurationFragment.setAlarm(alarm);
        fragmentManager.beginTransaction()
            .add(R.id.full_screen_fragment_container, shakeConfigurationFragment)
            .addToBackStack(null)
            .commit();
        shakeConfigurationFragment.setShakeConfigurationFragmentListener(new ShakeConfigurationFragment
            .ShakeConfigurationFragmentListener() {
                @Override
                public void onShakeConfigurationSetup(ShakeDetail shakeDetail) {
                    typeFragmentListener.getShakeChallenge(shakeDetail);
                    updateChallengeLayoutColor(ChallengeType.SHAKE);
                }
            });
    }
});
linearLayoutMoving.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        movingConfigurationFragment.setAlarm(alarm);
        fragmentManager.beginTransaction()
            .add(R.id.full_screen_fragment_container, movingConfigurationFragment)
            .addToBackStack(null)
            .commit();
        movingConfigurationFragment.setOnMovingConfigurationListener(new MovingConfigurationFragment
            .OnMovingConfigurationListener() {
                @Override
                public void onConfirm(MovingDetail movingDetail) {
                    typeFragmentListener.getMovingChallenge(movingDetail);
                    updateChallengeLayoutColor(MOVING);
                }
            });
    }
});

```

```

private void updateChallengeLayoutColor(ChallengeType newChallenge) {
    switch (currentChallengeType) {
        case DEFAULT:
            linearLayoutDefault.setBackgroundColor(getResources().getColor(R.color.challenge_layout_deactivate));
            break;
        case MATH:
            linearLayoutMath.setBackgroundColor(getResources().getColor(R.color.challenge_layout_deactivate));
            break;
        case SHAKE:
            linearLayoutShake.setBackgroundColor(getResources().getColor(R.color.challenge_layout_deactivate));
            break;
        case MOVING:
            linearLayoutMoving.setBackgroundColor(getResources().getColor(R.color.challenge_layout_deactivate));
            break;
    }
    switch (newChallenge) {
        case DEFAULT:
            linearLayoutDefault.setBackgroundColor(getResources().getColor(R.color.challenge_layout_activate));
            break;
        case MATH:
            linearLayoutMath.setBackgroundColor(getResources().getColor(R.color.challenge_layout_activate));
            break;
        case SHAKE:
            linearLayoutShake.setBackgroundColor(getResources().getColor(R.color.challenge_layout_activate));
            break;
        case MOVING:
            linearLayoutMoving.setBackgroundColor(getResources().getColor(R.color.challenge_layout_activate));
            break;
    }
    currentChallengeType = newChallenge;
}
}

```

+ Nhấn OK, trở về màn hình trước:

```

btnOK.setOnClickListener((view) -> {
    getSupportFragmentManager().popBackStack();
});

```

6. Màn hình cài đặt chế độ lắc:

- Tên người làm: Nguyễn Đào Hồng Nhung

a. Lựa chọn thêm hoặc giảm số lần lắc

```

this.textViewPlus25.setOnClickListener((v) -> {
    int newValue = shakeDetail.getNumberOfProblem() + 25;
    shakeDetail.setNumberOfProblem(newValue);
    numberPickerNumberOfProblem.setValue(newValue);
    textViewShakeNumberOfProblem.setText("Shake for " + newValue + " times");
});
this.textViewMinus25.setOnClickListener((v) -> {
    if (shakeDetail.getNumberOfProblem() <= 50) {
        return;
    }
    int newValue = shakeDetail.getNumberOfProblem() - 25;
    shakeDetail.setNumberOfProblem(newValue);
    numberPickerNumberOfProblem.setValue(newValue);
    textViewShakeNumberOfProblem.setText("Shake for " + newValue + " times");
});
this.textViewPlus50.setOnClickListener((v) -> {
    int newValue = shakeDetail.getNumberOfProblem() + 50;
    shakeDetail.setNumberOfProblem(newValue);
    numberPickerNumberOfProblem.setValue(newValue);
    textViewShakeNumberOfProblem.setText("Shake for " + newValue + " times");
});
this.textViewMinus50.setOnClickListener((v) -> {
    if (shakeDetail.getNumberOfProblem() <= 50) {
        return;
    }
    int newValue = shakeDetail.getNumberOfProblem() - 50;
    shakeDetail.setNumberOfProblem(newValue);
    numberPickerNumberOfProblem.setValue(newValue);
    textViewShakeNumberOfProblem.setText("Shake for " + newValue + " times");
});

```

+ Cộng thêm 25 lần lắc, trừ 25 lần lắc, cộng 50 lần lắc, trừ 50 lần lắc.

+ Nếu Number Picker có giá trị là 50 thì người dùng chỉ có thể tăng thêm số lần lắc chứ không thể giảm

b. Chọn mức độ lắc: Dễ, vừa, khó

```

this.radioButtonEasy.setOnClickListener((v) -> {
    shakeDetail.setDifficulty(EASY);
});
this.radioButtonModerate.setOnClickListener((v) -> {
    shakeDetail.setDifficulty(MODERATE);
});
this.radioButtonHard.setOnClickListener((v) -> {
    shakeDetail.setDifficulty(HARD);
});
this.buttonCancel.setOnClickListener((v) -> {
    getFragmentManager().popBackStack();
});

```

c. Hiển thị số lần lắc điện thoại tại text view Shake for...times

```

this.numberPickerNumberOfProblem.setOnValueChangedListener((picker, oldVal, newVal) -> {
    shakeDetail.setNumberOfProblem(newVal);
    textViewShakeNumberOfProblem.setText("Shake for " + newVal + " times");
});

```

7. Màn hình thử thách lắc:

- Tên người làm: Đào Phi Lạc

```
this.minInterval = 100;
switch(shakeDifficulty){
    case EASY:
        this.minForce = 20;
        break;
    case MODERATE:
        this.minForce = 40;
        break;
    case HARD:
        this.minForce = 60;
        break;
}
this.shakeDetector.configure(this.minInterval, this.minForce);
this.shakeDetector.start(new ShakeDetector.ShakeDetectorListener() {
    @Override
    public void onAccelerationChange(float x, float y, float z) { }
    @Override
    public void onShake(float force) {
        numberOfProblem--;
        if(numberOfProblem == 0){
            challengeActivity.challengeFinished();
        }
        else{
            textViewShakeNumberOfProblem.setText("Shake for " + numberOfProblem + " times");
        }
    }
    @Override
    public void onSupportDetection() { }
    @Override
    public void onNoSupportDetection() { }
    @Override
    public void onStopDetection() { }
});
```

- Khởi tạo một đối tượng ShakeDetector, đăng ký lắng nghe sự kiện onShake và giảm số lần lắc xuống. Nếu số lần lắc bằng 0 thì kết thúc thử thách

8. Màn hình cài đặt chế độ làm toán:

- Tên người làm: Trần Thị Thanh
- Các xử lý chức năng chính:

a. Set các giá trị cho TextView số phép toán và ví dụ mẫu cho mức độ khó đã chọn qua radio button.

```

        public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
            mathDetail.setNumberOfProblem(newVal);
            textViewMathNumberOfProblem.setText(newVal + " problem(s)");
        }
    });

    this.radioButtonEasy.setOnCheckedChangeListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            radioButtonClick(v);
            mathDetail.setDifficulty(EASY);
            textViewCalculationExample.setText(MathChallengeFragment.getCalculationExample(EASY));
        }
    });

    this.radioButtonModerate.setOnCheckedChangeListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            radioButtonClick(v);
            mathDetail.setDifficulty(MODERATE);
            textViewCalculationExample.setText(MathChallengeFragment.getCalculationExample(MODERATE));
        }
    });
}

```

b. Xác nhận, Hủy bỏ tùy chọn:

```

        this.buttonCancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getFragmentManager().popBackStack();
            }
        });

        this.buttonOk.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                listener.onMathConfigurationSetup(mathDetail);
                getFragmentManager().popBackStack();
            }
        });
}

```

- Button OK, gửi đối tượng mathDetail cho đối tượng lắng nghe khi math config được thiết lập.
- Button Cancel: trở về màn hình trước đó.

c. Cài đặt mặc định cho báo thức:

```

public void onAttach(Context context) {
    super.onAttach(context);
    this.context = context;
    this.databaseHandler = new DatabaseHandler(this.context);
    this.mathDetail = this.databaseHandler.getAlarmMathDetail(this.alarm.getIdAlarm());
    if(this.mathDetail == null){
        this.mathDetail = MathDetail.obtain(this.alarm.getIdAlarm());
    }
    this.listRadioButton = new ArrayList<>();
}

```

```

public static MathDetail obtain(int idAlarm) {
    return new MathDetail(idAlarm, MathDifficulty.MODERATE, numberOfProblem: 3);
}

public MathDetail(int idAlarm, int difficulty, int numberOfProblem) {
    this.idAlarm = idAlarm;
    this.difficulty = difficulty;
    this.numberOfProblem = numberOfProblem;
}

```

- Khi người chơi chọn thử thách làm toán, chế độ mặc định là 3 phép toán, mức độ khó là Moderate.

9. Màn hình thử thách làm toán:

- Tên người làm: Trần Thị Thanh
- Các xử lý chức năng chính:

a. Tạo phép tính mới:

```

private void generateCalculation(){
    random = new Random();
    int maxRange = 0;
    switch (mathDifficulty){
        case EASY:
            maxRange = 10;
            break;
        case MODERATE:
            maxRange = 20;
            break;
        case HARD:
            maxRange = 50;
            break;
        case INSANE:
            maxRange = 100;
            break;
        case NIGHTMARE:
            maxRange = 1000;
            break;
        case INFERNAL:
            maxRange = 10000;
            break;
    }

    for(int i=0;i<iNumCaculate;i++) {
        int a = random.nextInt(maxRange);
        int b = random.nextInt(maxRange);
        int c = random.nextInt(maxRange);

        String sCaculation= "(" + a + " x " + b + ")" + "+ c;
        int iResult=a*b+c;
        listCaculate.add(sCaculation);
        listResult.add(iResult);
    }
    getNextCaculate();
}

```

- Các số a, b, c được random theo 6 mức khác nhau (tùy vào cài đặt của người sử dụng)
- Sau khi random 3 số a, b, c. Phép tính theo công thức: $a*b+c$
- Kết quả của phép tính và phép tính cất listCaculate và listResult để gọi ra sử dụng sau này.
- Hàm generateCalculation() gọi tới hàm getNextCaculate() để gán phép tính vào textView câu hỏi hiển thị cho người dùng, đồng thời setText của textViewResult rỗng để người dùng nhập và kết quả cho phép tính sau.

```
private void getNextCaculate(){
    sUserResult="";
    textView_Question.setText(listCaculate.get(iNumberOfDoneCalculation));
    textView_Result.setText("");
}
```

b. Xác nhận, Xóa kết quả vừa nhập:

```
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        int iUserResult=Integer.parseInt(textView_Result.getText().toString());
        if(iUserResult==listResult.get(iNumberOfDoneCalculation)){
            iNumberOfDoneCalculation++;
            if(iNumberOfDoneCalculation==iNumCaculate){
                challengeActivity.challengeFinished();
            }
            else{
                getNextCaculate();
            }
        }
    }
});
```

- Khi người dùng bấm OK, kết quả phép tính người dùng nhập vào được so sánh với kết quả tính được khi random số ban đầu.
- Nếu đúng và số phép toán chưa đủ, hiển thị phép tính tiếp theo để người chơi trả lời. Nếu số phép tính đã đủ, thì ngừng báo thức.

```
buttonDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String sSysResult=sUserResult;
        sUserResult=sSysResult.substring(0,sUserResult.length()-1);
        textView_Result.setText(sUserResult);
    }
});
```

- Khi người dùng bấm Xóa, kết quả sẽ xóa trên text view kết quả.

c. Animation:


```

final Animation animRotate= AnimationUtils.loadAnimation(getContext(),R.anim.anim_rotate);
final Button btnSo0=view.findViewById(R.id.so0);
btnSo0.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        view.startAnimation(animRotate);
        sUserResult+=btnSo0.getText();
        textView_Result.setText(sUserResult);
    }
});

```

- Xoay các button từ 0 tới 9 khi người dùng bấm vào.

10. Màn hình cài đặt thử thách di chuyển

- Tên người làm: Đào Phi Lạc

```

distancePicker.setOnMeasurementChangeListener(new MeasurementPicker.OnMeasurementChange
    @Override
    public void onMeasurementChange(MeasurementPicker measurementPicker, Measurement
        movingDetail.setMeasurement(newValue);
        if(newValue == MeasurementPicker.Measurement.KILOMETER){
            linearLayoutAddMoreDistance.setVisibility(View.INVISIBLE);
        }
        else{
            linearLayoutAddMoreDistance.setVisibility(View.VISIBLE);
        }
    }
});
distancePicker.setOnValueChangeListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        movingDetail.setDistance(newVal);
    }
});
buttonCancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { fragmentManager.popBackStack(); }
});
buttonOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onMovingConfigurationListener.onConfirm(movingDetail);
        fragmentManager.popBackStack();
    }
});

```

- Đăng ký sự kiện OnMeasurementChange và OnValueChange của DistancePicker và cập nhật thuộc tính measurement và distance của đối tượng MovingDetail.
- Nút Cancel để trở về màn hình trước đó
- Nút OK gửi đối tượng MovingDetail đến đối tượng nào đã đăng ký lắng nghe sự kiện xác nhận cài đặt thử thách di chuyển.

11. Màn hình thử thách di chuyển

- Tên người làm: Đào Phi Lạc

```

private void start(){
    locationDetector.start(new MovingDetector.MovingDetectorListener() {
        @Override
        public void onMoved(float distance1, String furtherDetails) {
            distance -= distance1;
            if(distance <= 0){
                challengeActivity.challengeFinished();
            }
            else{
                textViewDistance.setText(String.format("%.3f", distance));
            }
        }

        @Override
        public void onFinished() {}
    });
}

```

- Khởi tạo một đối tượng LocationDetector, đăng ký lắng nghe sự kiện di chuyển của người dùng, trừ giá trị distance xuống. Khi giá trị của distance nhỏ hơn hoặc bằng 0 thì kết thúc thử thách.

Chương 4: Kết luận

1. Kết quả:

- ✓ Đã tạo được báo thức có reo chuông ở cả báo thức mặc định và báo thức bằng thử thách.
- ✓ Khi chưa hoàn thành báo thức, người dùng không thể thoát khỏi màn hình thử thách báo thức bằng bất cứ cách nào.
- ✓ Khi hoàn thành thử thách, báo thức ngừng reo, tắt màn hình thử thách.

2. Hạn chế:

- Tuy đã cài đặt được báo thức theo ý người sử dụng, tuy nhiên ứng dụng vẫn còn nhiều bug và chưa hoàn thiện hoàn toàn và thiếu sót.
Chúng em sẽ chỉnh sửa, bổ sung trong tương lai.